FAUNET: A program package for evalution of fault trees and networks

Platz, O.; Olsen, Jens V.

Publication date:
1976

Document Version
Publisher's PDF, also known as Version of record

Risø Report No. 348

Risø Report No. 348

Research Establishment Risø

FAUNET: A Program Package for Evaluation of Fault Trees and Network

by O. Platz and J. V. Olsen

September 1976

Sales distributors: Jul. Gjellerup, 87, Sølvgade, DK-1307 Copenhagen K, Denmark

Available on exchange from: Risø Library, Research Establishment Risø, DK-4000 Roskilde, Denmark

INIS Descriptors

ALGORITHMS
AVAILABILITY
COMPUTER CODES
FAULT TREE ANALYSIS
FORTRAN
PDP COMPUTERS
RELIABILITY

UDC 62.004.6 : 681.3.06

September 1976 Risø Report No. 348

FAUNET: A Program Package for Evaluation of Fault Trees and Networks

by

O. Platz and J. V. Olsen

Research Establishment Risø
Electronics Department

Abstract

A program package for the evaluation of fault trees and networks is described. The package, written in standard Fortran, is primarily designed for use on a minicomputer. Special efforts were therefore made to minimize storage space requirements.

The package contains programs for checking the fault tree input data for logical consistency and for providing various graphical representations of the tree. Furthermore, it contains programs for calculating minimal cut/path sets of the tree. A new technique automatically prunes the fault tree before the minimal set calculation. This pruning has in practice proved to considerably reduce computer time and storage space requirements.

The probability program calculates the availability of the system at various time points from the minimal sets and from failure and repair data for the components. The network program finds all path sets in a directed or undirected network where both links and nodes may fail. The path sets represent the structure function of the network. The minimal cut sets are found by subjecting the structure function to the minimal set programs. The terminal availabilities can be calculated by using the probability program.

ISBN 87-550-0418-0

CONTENTS

INTRODUCTION

Fault tree analysis (FTA) has become an indispensable tool for the reliability and safety analysis of complex systems. The basic concepts, techniques and applications are described in several papers and reports, e.g. (1 - 3). The state of the art is well represented by the contributions in (4) and (5).

Apart from being a valuable graphical tool for communication, FTA allows a quantitative assessment of system reliability and availability. In this respect, its applications are limited to systems where time sequence is not important, or at least of so little significance that it can be neglected. For systems where time sequence is essential, e.g. for systems where a passive standby is switched in when the main system fails, other techniques such as cause-consequence analysis (6, 7) are more appropriate. Even for systems of this kind, FTA often plays an important role in solving a part of the problem. For the example above, a FTA could be used to obtain the distribution of time to failure for the main system or the unavailability of the standby.

Computer programs for the analytic evaluation of fault trees are by now standard tools in the analysis of complex industrial systems. Many of the programs have been described in the literature (8 - 11). Most of them are written for large computers with ample in-core memory space. In this report we describe how procedures for fault tree evaluation can be implemented on a minicomputer (an 8K PDP8 under OS/8). This is made possible by an new storage-space-saving implementation of the Fussell Vesely (12) algorithm for calculating minimal sets, and by the development of a technique for automatic pruning (or modularization) of fault trees. Although the programs have been developed for use on a minicomputer because of its advantages with regard to interactive analysis and flexibility in handling of data files, they can, with minor modifications, be implemented on any computer that accepts FORTRAN.

The programs of the package fall into four main categories. In section 1 we describe the format of the fault tree input data and the programs under the common label TREE. The first of these programs performs various error checks on the input data. The function of the other programs is to provide graphical fault tree representations from the input data so that the tree can be represented in a form in which it is most easily surveyed.

Section 2 is concerned with the second category of programs under the general heading CUT. The function of these programs is to make a top-down calculation of minimal cut/path sets. One of the programs performs a pruning of the fault tree before it is subjected to the minimal set calculation. For most fault trees, the pruning will much reduce the number and/or the order of the minimal sets, thus making it possible to calculate reliability parameters with an increased accuracy in a shorter time. For the fault tree used as the main example in this report, pruning reduces the number of minimal cut sets from 11934 to 48.

Section 3 describes the UNA programs for reliability and availability calculations from the minimal sets and from failure, repair and test data for the components of the system. The programs accept components with either constant failure probabilities or exponentially distributed failure times. Repair times are either constant or exponentially distributed. Specifications of test intervals are also allowed.

Throughout sections 1 - 3 the fault tree for the Dresden-3 AC emergency power system is used as an example. It is taken from the report by Garrick, Gekler, et al. (13) where it was used to demonstrate a Monte Carlo code. Vesely (14) also used it. As a second example of a calculation of minimal cut sets, we use the fault tree for the High Pressure Injection System (HPIS) from Wash-1400, appendix 2, (15).

Network reliability has been extensively treated in the literature. Although of a general nature, much of the work has been aimed at communication networks. Surveys of work in this field are given in (16) and (17).

Methods for finding all paths between two nodes in a network have been presented by Fu and Yau (18), and Kroft (19). The terminal unavailability, i.e. the probability that the two nodes do not communicate at a given time point, can then be calculated as the probability that all paths between the two nodes are disconnected.

Jensen and Bellmore (20) developed an algorithm for determining all minimal cut sets for the communication between two nodes. The bounding technique of Esary and Proschan (21) was used to provide a lower limit to the terminal availability.

Nelson, Batts and Beadles (22) developed a method for the enumeration of all minimal cut sets between two nodes provided that the path sets were given. From either path or cut sets the terminal availability can be calculated by using the inclusion-exclusion technique (23).

Other methods for the calculation of various network reliability parameters are given in references 24-28.

The network program in the FAUNET package is described in section 4. For a network where links (components) and/or nodes can fail, the program TIENET calculates all paths between two specified nodes. The paths represent the success tree of the network. The Dual of this, the fault tree, is written on an output file. By subjecting the fault tree to a calculation with the CUT programs, all the minimal cut sets of the network can be found. If failure and repair distributions of links and nodes are known, the terminal unavailability can be found by using the UNA programs.

The network program can handle networks with both uni- and bi-directional links.

In section 4 the program is applied to two examples taken from references 20 and 22.

1. INPUT DATA AND TREE PROGRAMS

The format of the input data for the fault trees is illustrated by the data for the fault tree for the Dresden-3 system. The data are shown in fig. 1. They were obtained from appendix E in(13)where the tree structure was given in terms of logical FORTRAN statements. The first line in the file contains the name of the fault tree (FORTRAN format A6). Each of the following lines defines a gate in the tree in the form: gate type - gate number - number of gate inputs - list of gate inputs. The FORTRAN format is A1, 2I4. The gate type is specified as + or O for an OR gate, and as X or A for an AND gate. Events are indexed from 1 to 999 and gates are indexed from 1000 to 2000. The order in which the gates are defined in the file is immaterial. The $ sign in the last line denotes the end of the file.

For convenience in writing input files, the data can be written in free format with commas as delimiters. A program, FREE, then reads the input data in free format and produces an output file of data for either the Primal or the Dual tree in fixed format. The Dual tree is obtained from the Primal by converting AND gates to OR gates and vice versa, and by letting the basic events denote their logical inverses.

The TREE package consists of 4 programs:

TREECH, the error checking program, reads the input data in fixed format and lists the number of OR and AND gates and the number of basic events. It also produces a list of basic events and gates together with the the number of times they occur in the tree. It checks the tree to see if it is connected and to see if there are multiple definitions of a gate. If furthermore checks the tree for loops. Finally, it calculates the number of BICS

(Boolean Indicated Cut Sets (2)) and the maximum length of these for each gate. Apart from checking the data for errors, the functions performed by TREECH are helpful in assessing the size of the problem in the following calculation of minimal cut sets. As an option, the functions of TREECH can be performed on the Dual tree instead of the Primal. The information obtained is then pertinent to the path sets (tie sets) of the fault tree.

Figures 2 and 3, respectively, show the output produced by TREECH for the Primal and the Dual versions of the fault tree Dresden-3. The asterisks in the last line of fig. 2 just indicate that the number of BICS exceeded the format of the output statement. The number will be about $33 \cdot 10^{12}$.

TREEIN, the input module for the graphical programs, accepts input data in fixed format. The program asks for a specification of a top event (Gate). As a default it takes the gate in the second line of the input data (gate 1060 in fig. 1). With the selected top event as an ancestor, the program orders gates and events according to the generation in which they occur, and it allocates each component (gates and events) an initial set of coordinates for its position in the graph.

The structural description is stored in COMMON. Overflow data are left on a file for processing in a later pass.

TREEBA, the balancing module, accepts the tree description in COMMON from TREEIN and calculates new x-coordinates for positioning gates and events to give, as far as possible, a well balanced picture.

Initially, all components are placed as far to the left as possible. The tree is squeezed along the left edge of the picture. The balancing is an iterative process, moving the components from their initial left positions to the right on the plane. In a series of upwards and downwards scans the algorithm tries to establish a position of any gate name (output) and the inputs to the gate so that the output is placed above the central input, in the case of an odd number of inputs, or in the middle of the two central inputs in the case of an even number of inputs. The inputs are moved in the downgoing scan, and the outputs in the upgoing. The components do not move more than allowed by their neighbours. Only movements toward the right are permitted.

This process is repeated until no further movements occur. The result is still kept in COMMON while the output module is called.

TREEOU, the output module, draws the structure on the lineprinter, specifying gates by type, number and connection to inputs.

If there are more data than allowed on one sheet, the control is given back to TREEIN for another pass to evaluate the next part of the tree from the data stored in the overflow file.

If a gate occurs more than once in the input data, the subtree corresponding to this gate is developed the first time it appears as input (i.e. as near to the top gate as possible). The next time it occurs, e.g. in another branch or in a later generation, it is not expanded into its subtree but only referred to by the gate number. It is, nevertheless, possible to force the subtree corresponding to the gate to be "pressed" downwards in the tree by adding an extra line in the input data above the actual gate definition. This line should contain the gate number with no inputs. As gate type an ^ may be used to indicate that it is a subtree.

If a gate is "pressed" out of the tree in this way it will be processed separately and drawn on a separate sheet (e.g. gate 1055 in fig. 4).

2. CUT PROGRAMS

The package consists of 7 programs:

CUTSET	Input and gate contraction
CUTUP	Factoring, or modularisation, (bottom-up analysis)
CUTGO	Evaluation of minimal cut sets
CUTOFL	Overflow handling
CUTRED	Inter-block reduction
CUTOUT	Result calculation
CUTEV	Expansion of factorized minimal sets (optional)

Each of the first 6 programs calls the next in sequence by CHAIN.

CUTSET reads the input data in fixed format and asks for the highest order of sets wanted in the result together with a specification of the top event. As a default it uses the gate defined in the second line of the input file. The user may specify whether he requires a cut set or a tie set (path set) calculation of the fault tree. The program then contracts subsequent gates of the same type in the tree into a single gate, so that the tree structure becomes an alternating sequence of OR and AND gates. The data that are stored in COMMON are now ready for the next step:

CUTUP performs a bottom-up analysis by identifying primitive factors, i. e. a pair of basic events always occurring together in the same types of gate. Such a pair is replaced by a dummy event, here called a "complex event", while a description of this factor in terms of its type and inputs is written at the top of the result file. If, in the next pass, two complex events (or a complex event and a basic event) always appear as a pair in the same types of gate, they are combined into a new complex event and so on. This process continues until no more pairs are found. For each pass CUTUP prints "FACTORIZE". Finally CUTUP calls:

CUTGO, which identifies itself by the message "EVALUATE". CUTGO performs the actual evaluation of the cut sets/path sets. The evaluation is based on the Fussell Veseley algorithm (12). A description of the algorithm together with an application to a simple fault tree is also contained in (23).

All operations in CUTGO are performed in a single one-dimensional array placed in COMMON. The first part of the array contains the description of the factorized fault tree. Each gate is specified by type and inputs (gates, basic and/or complex events). For each gate there is a pointer to the array element where the information about the next gate begins. The remaining part of the array is reserved for cut set expansions. From now on this part is called the buffer.

The evaluation starts by replacing the top event with a list of its inputs. The list is placed in the buffer preceded by a pointer to indicate its length. The program then looks up the first gate in the buffer and expands it according to the Fussell Veseley algorithm. An AND gate will be expanded into a single and, in most cases, longer list of inputs. An OR gate will usually give rise to several new lists of the same length as the old one. The resulting lists are added to the lists already in the buffer together with pointers and the array elements which have been occupied by the old list are set free.

Every time the buffer is filled, i. e. no more free space is available at the end of the buffer, CUTGO performs a garbage collection to retrieve the free elements.

When no more free space is available, CUTGO enters the minimization phase. Here all cut sets, whether fully evaluated or not, are sorted according to size and compared for removal of all redundant sets. The message "MINIMIZE" is printed.

If some cut sets are removed, the program goes back to garbage collection followed by further expansions. If not, the overflow handler is called.

If the evaluation is finished (no cut sets contain gates as inputs) the overflow handler is called too.

CUTOFL, the overflow handler, starts with the message "OVERFLOW". This module extracts all partially evaluated sets as well as sets of order higher than that specified by the user as maximum. The sets are extracted from the array in COMMON and added to an overflow file on backing store.

The remaining sets are then added to the result file.

Finally, if there are any overflow data of lower order than the maximal, they are loaded into the buffer and control is given back to CUTGO to perform further evaluations.

If there are no more overflow data, or all the overflow data are of higher order than that specified as maximum, and all cut sets in the buffer have been fully evaluated, the control is given to:

CUTRED, which performs an inter-block minimization between the blocks in the result file that have been produced by several passes between CUTGO and CUTOFL. The message "REDUCE" is printed. CUTRED finally sorts all minimal sets in the result file according to order and input events.

The result file is a file containing:

1) calculation type (CUTSET/TIESET) and fault tree name,

2) a list of factors (definitions of complex events),

3) all minimal cut sets/tie sets found with order less than or equal to the maximal required.

CUTOUT is the final program that scans the result file and calculates the number of minimal sets of any order for the factorized tree, as well as the number of minimal sets for the original tree.

CUTEV, an independent program, may be called optionally. It accepts the result file as input, and as output it produces a list of the minimal sets for the original tree.

Examples

Example 1, Dresden-3

Figure 5 shows the questions and the messages printed by the computer and the answers typed by the operator under the execution of the Dresden-3 fault tree. The first line is a run command to the operating system. In the next lines the operator specifies that he requires a cut set calculation on the input data existing as a disk (DSK) file DRESD3. He makes certain that the

cut sets are calculated to any order by typing 999 as answer to the question about the highest order wanted. As a default, the computer selects gate 1060 as top event.

After having made several passes through minimization, overflow and evaluate stages, the output module CUTOUT prints the number and the order of the minimal cut sets for the factorized tree and for the original tree. For the Dresden-3 fault tree the modularization technique causes a reduction of the total number of minimal cut sets from 11934 to 48.

The result file is listed in fig. 6. The first line contains the name of the fault tree together with a specification of the type (cut/tie) of the sets. The following lines contain the definitions of the complex events, which are indexed from 999 and downwards; -1 indicates an OR gate, -2 an AND gate. The second line in the file thus shows that complex event no. 999 is an OR gate with the basic events 50 and 51 as inputs.

The 48 minimal cut sets follow after a mark (-959 here). The first number in each line is just an indication of the number of data in the line. If the complex events in these cut sets were expanded (e.g. by CUTEV) according to their definitions, the 11934 minimal sets for the original tree would be obtained.

Figures 7-8 show the calculation of the minimal tie sets for the Dresden-3 fault tree. The complex events in fig. 8 are, of course, identical to those in fig. 6, except for the fact that OR gates in the Primal tree are converted to AND gates in the Dual tree. The total number of path sets are 8 both for the factorized tree and for the original tree, but they are of considerably lower order in the factorized tree.

Example 2, HPIS

Figures 9-13 show the input data, the outputs of the checking program, and the minimal cut set programs for the fault tree of the High Pressure Injection System taken from appendix 2 of Wash-1400. The 8179 minimal cut sets of the original tree are reduced to 52 minimal sets in the factorized tree.

It was not possible to calculate all the minimal path sets of the HPIS fault tree. As seen from fig. 11, there are 248832 BICS (or rather BIPS, Boolean Indicated Path Sets) of maximal length 174. The number, and size, of the actual minimal path sets, even for the factorized tree, exceeded the capacity of the PDP8 minicomputer.

3. PROBABILITY PROGRAMS, UNA

The UNA programs calculate system unavailability from the minimal sets of the factorized tree when failure, repair and test data are given for the components of the system. The components are assumed to be statistically independent.

In their present form the programs accept exponentially distributed failure times and repair times that are either constant or exponentially distributed. They also accept constant failure probabilities for components with a fixed probability of failure per demand. For the purpose of treating components where failures are only detected at tests, the programs furthermore allow the specification of constant test intervals.

In order to minimize the requirements to core storage space, the programs perform a dynamic allocation of storage space similar to that used in the CUT programs. Only a single one-dimensional array is used. In the following the array that is placed in COMMON is called the internal buffer.

UNAVA1, the main program, administrates the subroutine calls. It first calls:

UNAIN1, which reads the basic failure, repair and test data from the input data file. The information is stored in the internal buffer. The program then reads the complex events (i. e. type, number and inputs) from the cut/tie set file, the result file from the CUT programs. This information is also placed in the internal buffer. Finally, the program reads the minimal sets from the same file and stores them in a direct access disk file, the external buffer. The control is then given back to the main program which asks the operator to specify the time points at which he requires the calculations performed and the number of inclusion-exclusion terms, he needs in the calculation of the system unavailability. The main program then calls:

UNASU1, which calculates the unavailability of the components from their failure data. If a component has an exponential failure distribution with failure rate λ and exponential repair distribution with mean repair time r, the unavailability as a function of time t is given by the well known expression

$$u(t) = \frac{\lambda}{\lambda + 1/r} \left(1 - e^{-(\lambda + 1/r)t}\right),$$

when the component was operating at time 0.

If the component has an exponential failure distribution with failure rate λ and constant repair time r the unavailability is given by (29)

$$u(t) = 1 - \sum_{j=o}^{[t/r]} \frac{\lambda^j (t-rj)^j e^{-\lambda(t-rj)}}{j!} ,$$

where $[t/r]$ denotes the greatest integer contained in t/r.

For a component with an exponential failure distribution $1 - e^{-\lambda t}$ and constant repair time r, and subjected to periodic tests with a test interval T, so that failures are revealed and repaired at tests only, the point unavailability is easily found to be given by the expression:

$$u(t) = \begin{cases} 1-e^{-\lambda t} & \text{for} \quad t < T \\ (1-a)(1-e^{-\lambda t'}) + a & \text{for} \quad t \geq T \quad \text{and} \quad t' \leq r \\ (1-a)(1-e^{-\lambda t'}) & \\ \quad + a(1-e^{-\lambda(t'-r)}) & \text{for} \quad t \geq T \quad \text{and} \quad t' > r \end{cases}$$

here $t' = t$ modulo T, and $a = 1-e^{-\lambda T}$. It is assumed that $r \leq T$.

After having calculated the unavailabilities of all the components the program stores the results in the internal buffer and the next subroutine is called.

UNACP1 calculates the probabilities of the complex events from the probabilities of the basic events, i. e. the unavailabilities of the components. If the complex event 1 is an OR gate with input events (basic or complex) m and n, the probability p_1 of the complex event is given by

$$p_1 = p_m + p_n - p_m \cdot p_n ,$$

where p_m and p_n are the probabilities of the statistically independent input events. For an AND gate the corresponding expression is given by

$$p_1 = p_m \cdot p_n .$$

Having calculated the probabilities of all complex events, the program stores the results in the internal buffer and the main program calls:

UNAFI1, which performs the final calculation of system unavailability from the minimal sets of the factorized tree and from the probabilities of basic and complex events.

The program calculates the probability of the union of the minimal sets by using the inclusion-exclusion technique(23). In its present form the program is able to include the three first terms in the expansion.

Example, Dresden-3

The failure data for the Dresden-3 fault tree are taken from table 2.10 of the Holms and Narver report(13). Figure 14 shows the data written as a free format input file for the UNA programs. Each line in the file consists of component number, calculation type, number of succeeding data inputs, failure data, mean repair time and test interval.

Generally, calculation type 1 denotes a constant failure probability; type 2, exponential failure and repair distributions; while types 3 and 4 denote exponential failure distribution and constant repair time. In the case of type 4, failures are only detected at tests.

Failure data are either constant probabilities (calculation type = 1), or failure rates. In both cases the program multiplies the number in the input file by 10^{-6}.

In the HN report, the Dresden-3 fault tree was run with the SAFTE Monte Carlo simulation code. The failure rates given for the components were used, but the components were assumed to be non-repairable. Vesely(14) also used this fault tree to illustrate the KITT computer codes. Because of the amount of cut sets in the fault tree, Vesely based his calculations on the minimal path sets. A KITT-1 run was made with the 8 minimal path sets found. The failure rates used were those given in the HN report. Vesely also made a KITT-2 run in order to obtain results for the repairable system. The components were assumed to have constant repair times, the average repair times given in the HN report were used and the standard deviations associated with the actual normal repair distributions were ignored. Vesely obtained the exact answer by using all inclusion-exclusion terms for the minimal path sets. In general, it is necessary to use all terms in a probability calculation based on minimal path sets in order to obtain sufficient accuracy when the component failure probabilities are small.

Figures 15-16 show the results of calculations with the UNA programs for the same cases as considered by Vesely. In contrast to his calculations, the calculations here were based on minimal cut sets, i.e. the 48 minimal cut sets of the reduced fault tree. Two inclusion-exclusion terms were calculated. The terms provide upper and lower bounds to the exact probabilities. As seen from the figures, the first term alone gives a sufficiently accurate answer for most of the time range considered.

To facilitate comparison with the results from (13) and (14), figure 15 shows a calculation for the non-repairable case with time points corresponding to an integer number of weeks. The results are in good agreement with the graphs in (13) and (14).

For the repairable case, the results are given in figure 16 for the time points corresponding to those calculated in (14). The results in figure 16 differ slightly in the second decimal point from those in (14). The error is presumably due to the small word length in the PDP8 minicomputer (mantissa = 24 bits).

4. NETWORK PROGRAM: TIENET

For a network with bi- or uni-directional links (components) the program calculates all paths between two nodes specified by the user. The paths represent the "success" tree of the network. The Dual of this, the fault tree, is written on an output file. The data are acceptable in this form as input data for the CUT programs.

The algorithm used here is identical in principle to that given by Kroft (19), but the implementation differs. Kroft's algorithm was based on a description of the nodes only. We are interested in obtaining the fault tree of the system in terms of its representation by minimal cut sets where the basic events represent failures of components or nodes, or both. It is therefore necessary to use a description of the network in terms of its links as well as of its nodes.

Example 2 in reference 22 is used to illustrate the performance of TIENET. The network is shown in figure 17. As seen from the figure, all links are uni-directional. The component numbers are those given in 22. The nodes have been given the numbers shown in the figure.

The input data file is shown in figure 18. Each line in the file consists of: component number - input node - output node. The component numbers in this file are all negative, which signifies that the links are uni-directional. Bi-directional links are written with positive component numbers in the input data. TIENET then replaces each bi-directional link with its two uni-directional equivalents.

Two internal lists, a component list and a node list, figure 19, are made by TIENET from the input data. Both lists are placed in the same one-dimensional array. The component list is a list of component descriptions arranged according to component number. Each component description contains: component number - input node - output node. In the node

list each node is described in terms of the next components that can be reached from it by a flow originating in the node. Each node description consists of: a pointer L to the first element in the next node description - node number - N, the total number of succeeding component pointers - an element with a pointer (IPREV) to the previous node encountered in the instant path search - a pointer (IFLAG) pointing towards one of the succeeding component pointers, the one representing the component under investigation in the instant path search (IFLAG$\neq$ 0 indicates that the node has been encountered previously in the path search) - and a series of N component pointers (each pointer pointing towards the corresponding component description in the component list).

A flowchart of the search algorithm used in TIENET is given in figure 20. The algorithm searches out all paths between the start node and the end node, avoiding paths with loops.

The output file from TIENET is shown in figure 21. If the output-file is subjected to a "cut set" calculation by the CUT programs the minimal cut sets of the network are found and written out in standard form, figure 22. If the file is subjected to a "tie set" calculation, the minimal path sets of the network are written out, figure 23. The expanded minimal cut and path sets are identical to those found in (22).

As a second example we use the network given in figure 1 of reference 20. It is reproduced here in figure 24. The input data file is shown in figure 25.

Figure 26 shows three runs of TIENET for the paths between nodes 1 and 8; one for nodes only, one for components only, and one for both nodes and components. When the three output files are used as inputs to the CUT programs, the results in figures 27 - 29 are obtained. The minimal cut sets for components only are identical to those found in (20). The number of minimal cut sets differs in the three cases because of the difference in vulnerability of the network to node failures and to link failures.

The number of minimal path sets is, of course, the same in the three cases. The only difference lies in the way in which they are specified. The path sets for components only are shown in figure 30.

REFERENCES

1) R.E. Barlow and P. Chatterjee, Introduction to Fault Tree Analysis, ORC 73-30, AD-774 072 (1973) 48 pp.

2) J.B. Fussell, Fault Tree Analysis - Concepts and Techniques. In: Generic Techniques in Systems Reliability Assessment. Edited by E.J. Henley and J.W. Lynn. Nato Advanced Study Institute. (Nordhoff, Leiden, 1974).

3) H.E. Lambert, Fault Trees for Decision Making in Systems Analysis. Ph. D. Thesis. URCL - 51829 (1975) 351 pp.

4) Reliability and Fault Tree Analysis. Theoretical and Applied Aspects of System Reliability and Safety Assessment. Conference held at the University of California, Berkeley, 3-7 September 1974. Edited by R.E. Barlow, J.B. Fussell, and N.D. Singpurwalla. (Society for Industrial and Applied Mathematics, Philadelphia, Pen., 1975) 927 pp.

5) J.B. Fussell, G.J. Powers and R.G. Bennetts, Fault Trees - A State of the Art Discussion. IEEE Trans. Reliab. R-23. (1974) 51-55.

6) D. Nielsen, Use of Cause-Consequence Charts in Practical Systems Analysis. In: Reliability and Fault Tree Analysis. Theoretical and Applied Aspects of System Reliability and Safety Assessment. Conference held at the University of California, Berkeley, 3-7 September 1974. Edited by R.E. Barlow, J.B. Fussell, and N.D. Singpurwalla. (Society for Industrial and Applied Mathematics, Philadelphia, Penn., 1975) 849-880.

7) J.R. Taylor, Sequential Effects in Failure Mode Analysis. In: Reliability and Fault Tree Analysis. Theoretical and Applied Aspects of System Reliability and Safety Assessment. Conference held at the University of California, Berkeley, 3-7 September 1974. Edited by R.E. Barlow, J.B. Fussell, and N.D. Singpurwalla (Society for Industrial and Applied Mathematics, Philadelphia, Penn., 1975) 881-894.

8) W.E. Vesely and R.E. Narum, PREP and KITT: Computer Codes for the Automatic Evaluation of a Fault Tree. IN-1349 (1970) 188 pp.

9) A.G. Colombo, CADI, a Computer Code for System Availability and Reliability Evaluation. EUR - 4940e (1973) 43 pp.

10) J.B. Fussell, E.B. Henry, N.H. Marshall, MOCUS: a Computer Program to obtain Minimal Sets from Fault Trees. ANCR - 1156 (1974) 42 pp.

11) P.K. Pande, M.E. Spector, P. Chatterjee, Computerized Fault Tree Analysis: TREEL and MICSUP. ORC 75-3, AD-A 010 146 (1975) 56 pp.

12) J.B. Fussell and W.E. Vesely, A new Methodology for Obtaining Cut Sets for Fault Trees. Trans. Am. Nucl. Soc. 15 (1972) 262-263.

13) B.J. Garrick, W.C. Gekler et al., Reliability Analysis of Nuclear Power Plant Protective Systems. HN-190 (1967) 600 pp.

14) W.E. Vesely, Analysis of Fault Trees by Kinetic Tree Methods. IN-1330 (1969) 94 pp.

15) Reactor Safety Study. An Assessment of Accident Risks in U.S. Commercial Nuclear Power Plants. Vol. 1-9. WASH-1400, NUREG-75/014 (1975).

16) H. Frank and I.T. Frisch, Analysis and Design of Survivable Networks. IEEE Trans. Commun. Technol. COM-18 (1970) 501-519.

17) R.S. Wilkov, Analysis and Design of Reliable Computer Networks. IEEE Trans. Commun. COM-20 (1972) 660-678.

18) Y. Fu and S.S. Yau, A Note on the Reliability of Communication Networks. J. Soc. Ind. Appl. Math., 10 (1962) 469-474.

19) D. Kroft, All Paths Through a Maze. Proc. IEEE 55 (1967) 88-90.

20) P.A. Jensen and M. Bellmore, An Algorithm to Determine the Reliability of a Complex System. IEEE Trans. Reliab. R-18 (1969) 169-174.

21) J.D. Esary and F. Proschan, Coherent Structures of Non-identical Components. Technometrics 5 (1963) 191-209.

22) A.C. Nelson, J.R. Batts and R.L. Beadles, A Computer Program for Approximating System Reliability. IEEE Trans. Reliab. R-19 (1970) 61-65.

23) R.E. Barlow and F. Proschan, Statistical Theory of Reliability and Life Testing, Probability Models. (Holt, Rinehart and Winston, Inc., New York, 1975) 290 pp.

24) K.B. Misra and T.S.M. Rao, Reliability Analysis of Redundant Networks. Using Flow Graphs. IEEE Trans. Reliab. R-19 (1970) 19-24.

25) K. B. Misra, An Algorithm for the Reliability Evaluation of Redundant Networks, IEEE Trans. Reliab. R-19 (1970) 146-151.

26) E. Hänsler, A Fast Recursive Algorithm to Calculate the Reliability of a Communication Network. IEEE Trans. Commun. COM-20 (1972) 637-640.

27) E. Hänsler, G. K. McAuliffe, R. S. Wilkow, Exact Calculation of Computer Network Reliability. Networks 4 (1974) 95-112.

28) O. Wing and P. Demetriou, Analysis of Probabilistic Network. IEEE Trans. Commun. Syst. CS-12 No. 3 (9164) 38-40.

29) R. E. Barlow and F. Proschan, Mathematical Theory of Reliability. (John Wiley and Sons, New York, 1965) 256 pp.

```
DRESD3
X1060   210591058
X1059   210451053
+1053   210511052
+1051   21050  55
+1050   210481049
+1049   2  53  54
+1048   2  521047
+1047   210461057
+1046   2  51  50
+1052   2  56  57
+1045   210431044
+1044   2  48  49
+1043   2  471042
+1042   210401041
+1041   2  45  46
+1040   2  441039
+1039   210381055
+1038   2  42  43
X1058   210551057
X1057   3103210341037
+1037   210251036
+1036   2  40  41
+1034   210201033
+1033   2  36  37
+1032   210311056
X1056   210281030
+1028   210171027
+1027   2  30  31
+1017   2  181015
+1030   210091029
+1029   2  32  33
+1031   2  34  35
X1055   3101910221026
+1026   210231025
+1025   3  2910351024
+1035   2  38  39
+1024   2  27  28
+1023   2  25  26
+1022   210201021
+1021   2  21  22
+1020   2  23  24
+1019   210181054
+1018   2  19  20
X1054   210111016
+1011   2  121010
+1010   2  111009
+1009   2  101008
+1008   210061007
+1007   2   8   9
+1006   2   2   7
+1016   2  171015
+1015   2  161014
+1014   210121013
+1013   2  14  15
+1012   2  131005
+1005   210031004
+1004   2   5   6
+1003   210011002
+1002   2   3   4
+1001   2   1   2
$
```

Fig. 1. Input data for the Dresden-3 fault tree.

```
CHECK OF INPUT DATA FOR :      DRESD3 (PRIMAL)

GATE 1060 OCCURS ON LEFT SIDE BUT NOT ON RIGHT

NO OF DIFFERENT OR - GATES =       53
NO OF DIFFERENT AND - GATES =       7

TOTAL NO OF DIFFERENT GATES =      60
NO OF DIFFERENT EVENTS =           57

EVENTS       1   2   3   4   5   6   7   8   9  10  11  12
REPLICATION  4   8   4   4   4   4   4   4   4   4   2   2

EVENTS      13  14  15  16  17  18  19  20  21  22  23  24
REPLICATION  4   4   4   4   2   2   2   2   2   2   4   4

EVENTS      25  26  27  28  29  30  31  32  33  34  35  36
REPLICATION  2   2   4   4   4   2   2   2   2   2   2   2

EVENTS      37  38  39  40  41  42  43  44  45  46  47  48
REPLICATION  2   4   4   2   2   1   1   1   1   1   1   1

EVENTS      49  50  51  52  53  54  55  56  57
REPLICATION  1   1   1   1   1   1   1   1   1
```

GATE	BICS	MAX LENGTH	REPLICATION
1001	2	1	1
1002	2	1	1
1003	4	1	1
1004	2	1	1
1005	6	1	1
1006	2	1	1
1007	2	1	1
1008	4	1	1
1009	5	1	2
1010	6	1	1
1011	7	1	1
1012	7	1	1
1013	2	1	1
1014	9	1	1
1015	10	1	2
1016	11	1	1
1017	11	1	1
1018	2	1	1
1019	79	2	1
1020	2	1	2
1021	2	1	1
1022	4	1	1
1023	2	1	1
1024	2	1	1
1025	3	1	2
1026	7	1	1
1027	2	1	1
1028	13	1	1
1029	2	1	1
1030	7	1	1
1031	2	1	1
1032	93	2	1
1033	2	1	1
1034	4	1	1
1035	2	1	1
1036	2	1	1
1037	7	1	1
1038	2	1	1
1039	2214	4	1
1040	2215	4	1
1041	2	1	1
1042	2217	4	1
1043	2218	4	1
1044	2	1	1
1045	2220	4	1
1046	2	1	1
1047	2606	4	1
1048	2607	4	1
1049	2	1	1
1050	2609	4	1
1051	2610	4	1
1052	2	1	1
1053	2612	4	1
1054	77	2	1
1055	2212	4	2
1056	91	2	1
1057	2604	4	2
1058	5760049	8	1
1059	5798642	8	1
1060**********		16	

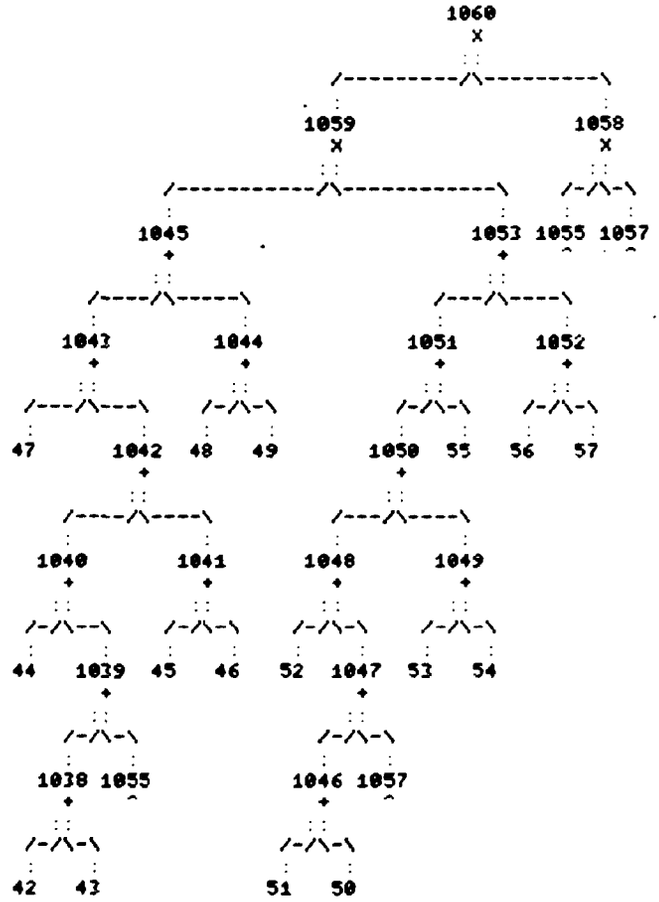
Fig. 2. TREECH output for the Dresden-3 fault tree.

```
CHECK OF INPUT DATA FOR       DRESD3 (  DUAL)

GATE 1060 OCCURS ON LEFT SIDE BUT NOT ON RIGHT

NO OF DIFFERENT OR - GATES =       7
NO OF DIFFERENT AND - GATES =     53

TOTAL NO OF DIFFERENT GATES =     60
NO OF DIFFERENT EVENTS =          57

EVENTS       1   2   3   4   5   6   7   8   9  10  11  12
REPLICATION  4   8   4   4   4   4   4   4   4   4   2   2

EVENTS      13  14  15  16  17  18  19  20  21  22  23  24
REPLICATION  4   4   4   4   2   2   2   2   2   2   4   4

EVENTS      25  26  27  28  29  30  31  32  33  34  35  36
REPLICATION  2   2   4   4   4   2   2   2   2   2   2   2

EVENTS      37  38  39  40  41  42  43  44  45  46  47  48
REPLICATION  2   4   4   2   2   1   1   1   1   1   1   1

EVENTS      49  50  51  52  53  54  55  56  57
REPLICATION  1   1   1   1   1   1   1   1   1
```

GATE	BICS	MAX LENGTH	REPLICATION
1001	1	2	1
1002	1	2	1
1003	1	4	1
1004	1	2	1
1005	1	6	1
1006	1	2	1
1007	1	2	1
1008	1	4	1
1009	1	5	2
1010	1	6	1
1011	1	7	1
1012	1	7	1
1013	1	2	1
1014	1	9	1
1015	1	10	2
1016	1	11	1
1017	1	11	1
1018	1	2	1
1019	2	13	1
1020	1	2	2
1021	1	2	1
1022	1	4	1
1023	1	2	1
1024	1	2	1
1025	1	5	2
1026	1	7	1
1027	1	2	1
1028	1	13	1
1029	1	2	1
1030	1	7	1
1031	1	2	1
1032	2	15	1
1033	1	2	1
1034	1	4	1
1035	1	2	1
1036	1	2	1
1037	1	7	1
1038	1	2	1
1039	4	15	1
1040	4	16	1
1041	1	2	1
1042	4	18	1
1043	4	19	1
1044	1	2	1
1045	4	21	1
1046	1	2	1
1047	4	17	1
1048	4	18	1
1049	1	2	1
1050	4	20	1
1051	4	21	1
1052	1	2	1
1053	4	23	1
1054	2	11	1
1055	4	13	2
1056	2	13	1
1057	4	15	2
1058	8	15	1
1059	8	23	1
1060	16	23	

Fig. 9. TREECH output for the Dual of the Dresden-3 fault tree.

1 SYSTEM: DRESD3 PART: 1



1

Fig. 4. Dresden-3 fault tree.

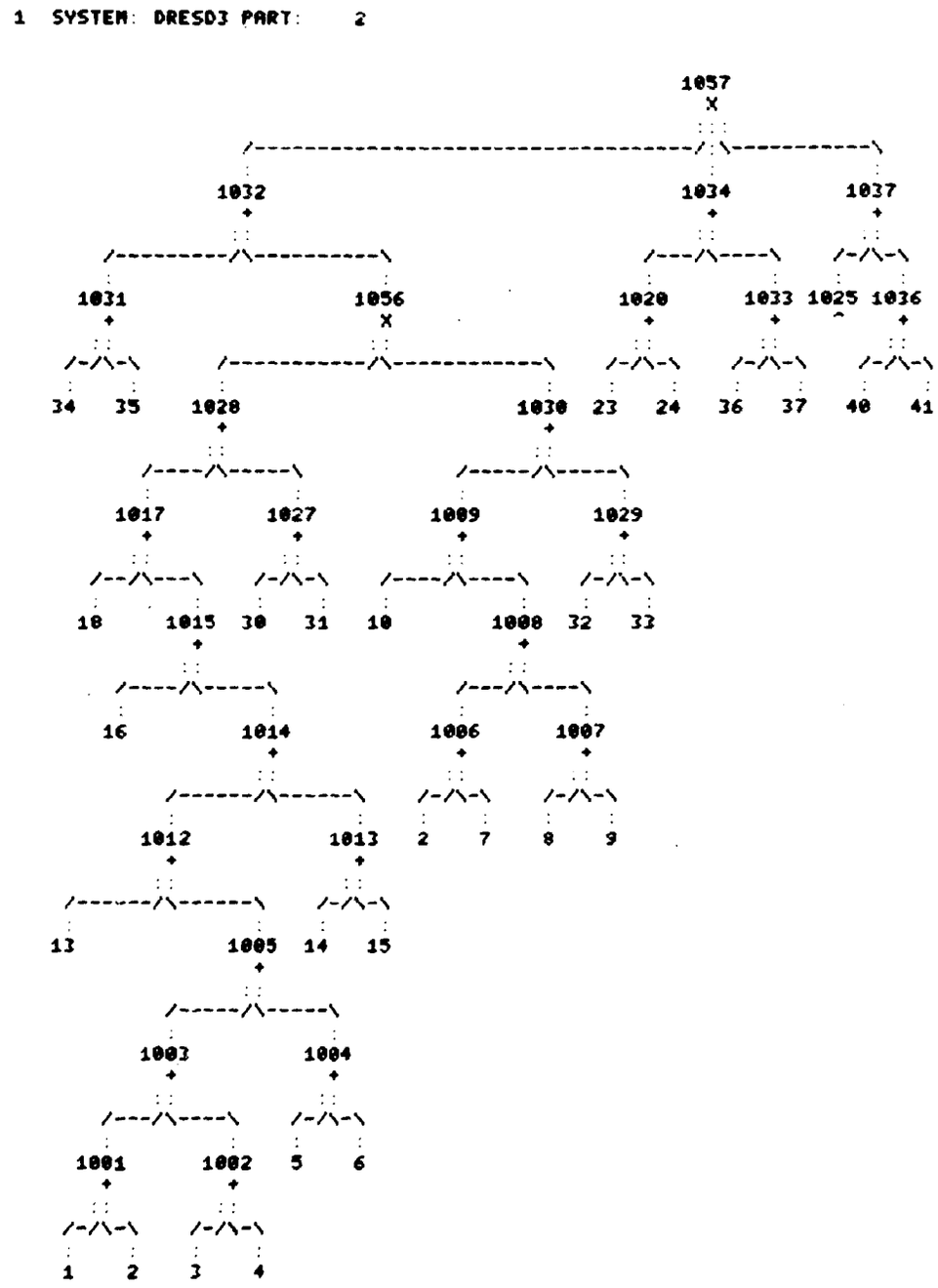


Fig. 4. Dresden-3 fault tree.

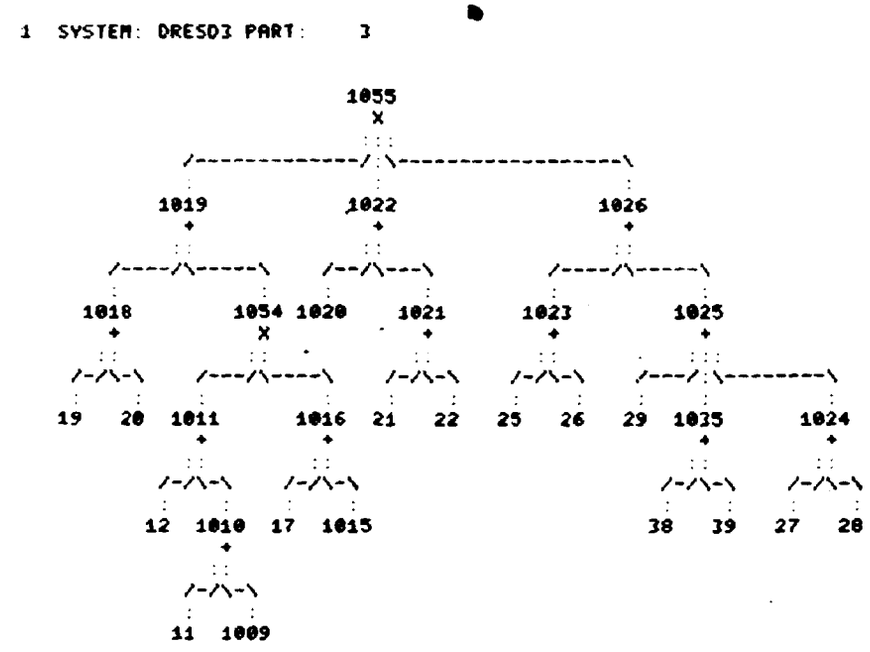


Fig. 4. Dresden-3 fault tree.

```
 R CUTSET
CUTSET OR TIESET: CUTSET
CUTSET OF:
 FILE DSK0: DRESD3
SYSTEM: DRESD3
HIGHEST ORDER WANTED = 999
 TOP GATE:
GATE: 1060 SELECTED AS TOP
FACTORIZE
EVALUATE
MINIMIZE
MINIMIZE
MINIMIZE
MINIMIZE
MINIMIZE
MINIMIZE
MINIMIZE
MINIMIZE
MINIMIZE
MINIMIZE
OVERFLOW
EVALUATE
MINIMIZE
MINIMIZE
MINIMIZE
MINIMIZE
OVERFLOW
EVALUATE
MINIMIZE
MINIMIZE
MINIMIZE
OVERFLOW
EVALUATE
MINIMIZE
MINIMIZE
MINIMIZE
OVERFLOW
EVALUATE
MINIMIZE
MINIMIZE
OVERFLOW
REDUCE
OUTPUT


RESULT OF DRESD3

REDUCED CUTSET:
       1.  SETS OF ORDER  3
       4.  SETS OF ORDER  4
      13.  SETS OF ORDER  5
      19.  SETS OF ORDER  6
      10.  SETS OF ORDER  7
       1.  SETS OF ORDER  8
------------
      48.

EVALUATED CUTSET:
      10.  SETS OF ORDER  3
     428.  SETS OF ORDER  4
    2816.  SETS OF ORDER  5
    5464.  SETS OF ORDER  6
    3024.  SETS OF ORDER  7
     192.  SETS OF ORDER  8
------------
   11934.
```

Fig. 5 Teletype output during calculation of minimal cut sets of the Dresden-3 fault tree.

```
CUTSET OF DRESD3
     4    -1   999    50    51
     4    -1   998    42    43
     4    -1   997    34    35
     4    -1   996    27    28
     4    -1   995    23    24
     4    -1   994    19    20
     4    -1   993     7     8
     4    -1   992     1     3
     4    -1   [illegible]  52    53
     4    -1   [illegible]  44    45
     4    -1   989    29    38
     4    -1   988    36    37
     4    -1   987    21    22
     4    -1   986     9    10
     4    -1   985     4     5
     4    -1   984    54    55
     4    -1   983    46    47
     4    -1   982    39   989
     4    -1   981    32    33
     4    -1   980    11    12
     4    -1   979     6    13
     4    -1   978    56    57
     4    -1   977    48    49
     4    -1   976    40    41
     4    -1   975    25    26
     4    -1   974   986   993
     4    -1   973    14    15
     4    -1   972   978   984
     4    -1   971   977   983
     4    -1   970   982   996
     4    -1   969    16   973
     4    -1   968   972   991
     4    -1   967   971   990
     4    -1   966   969   979
     4    -1   965   968   999
     4    -1   964   967   998
     4    -1   963   966   985
     4    -1   962   963   992
     4    -1   961    18    30
     4    -1   960    31   961
-959
     3     2   970   995
     4     2   970   987   988
     4     2   975   976   995
     4   962   970   974   995
     4   970   994   995   997
     5     2   975   976   987   988
     5    17   960   970   974   995
     5    17   970   974   995   997
     5    17   970   980   995   997
     5   960   970   974   994   995
     5   960   970   981   994   995
     5   962   970   974   987   988
     5   962   970   980   981   995
     5   962   970   980   995   997
     5   962   970   981   994   995
     5   962   974   975   976   995
     5   970   987   980   994   997
     5   975   976   994   995   997
     6    17   960   970   974   987   988
     6    17   960   970   980   981   995
     6    17   960   974   975   976   995
     6    17   970   974   987   988   997
     6    17   970   980   987   988   997
     6    17   974   975   976   995   997
     6    17   975   976   980   995   997
     6   960   970   974   987   988   994
     6   960   970   981   987   988   994
     6   960   974   975   976   994   995
     6   960   975   976   981   994   995
     6   962   970   980   981   987   988
     6   962   970   980   987   988   997
     6   962   970   981   987   988   994
     6   962   974   975   976   987   988
     6   962   975   976   980   981   995
     6   962   975   976   980   995   997
     6   962   975   976   981   994   995
     6   975   976   987   980   994   997
     7    17   960   970   980   981   987   988
     7    17   960   974   975   976   987   988
     7    17   960   975   976   980   981   995
     7    17   974   975   976   987   988   997
     7    17   975   976   980   987   988   997
     7   960   974   975   976   987   988   994
     7   960   975   976   981   987   988   994
     7   962   975   976   980   981   987   988
     7   962   975   976   980   987   988   997
     7   962   975   976   981   987   988   994
     8    17   960   975   976   980   981   987   988
```

Fig. 6. Result file for the Dresden-3 fault tree.

```
 R CUTSET
CUTSET OR TIESET: TIESET
TIESET OF:  DSK
 FILE DSK@: DRESD3
SYSTEM: DRESD3
HIGHEST ORDER WANTED = 999
 TOP GATE:
GATE: 1060 SELECTED AS TOP
FACTORIZE
EVALUATE
MINIMIZE
MINIMIZE
OVERFLOW
REDUCE
OUTPUT


 RESULT OF DRESD3

 REDUCED TIESET
        4  SETS OF ORDER  2
        4  SETS OF ORDER  4
 ------------
        8.

 EVALUATED TIESET:
        2  SETS OF ORDER  4
        2  SETS OF ORDER  7
        2  SETS OF ORDER  9
        1  SETS OF ORDER 13
        1  SETS OF ORDER 15
 ------------
        8
```

Fig. 7. Teletype output during calculation of minimal path sets of the Dresden-3 fault tree.

```
TIESET OF DRESD3
     4   -2  999   50   51
     4   -2  998   42   43
     4   -2  997   34   35
     4   -2  996   27   28
     4   -2  995   23   24
     4   -2  994   19   20
     4   -2  993    7    8
     4   -2  992    1    3
     4   -2  991   52   53
     4   -2  990   44   45
     4   -2  989   29   38
     4   -2  988   36   37
     4   -2  987   21   22
     4   -2  986    9   10
     4   -2  985    4    5
     4   -2  984   54   55
     4   -2  983   46   47
     4   -2  982   39  989
     4   -2  981   32   33
     4   -2  980   11   12
     4   -2  979    6   13
     4   -2  978   56   57
     4   -2  977   48   49
     4   -2  976   40   41
     4   -2  975   25   26
     4   -2  974  986  993
     4   -2  973   14   15
     4   -2  972  978  984
     4   -2  971  977  983
     4   -2  970  982  996
     4   -2  969   16  973
     4   -2  968  972  991
     4   -2  967  971  990
     4   -2  966  969  979
     4   -2  965  968  999
     4   -2  964  967  998
     4   -2  963  966  985
     4   -2  962  963  992
     4   -2  961   18   30
     4   -2  960   31  961
-959
     2  970  975
     2  970  976
     2  987  995
     2  988  995
     4    2   17  962  994
     4    2  960  962  997
     4    2  974  980  994
     4    2  974  981  997
```

Fig. 8. Minimal path sets of the Dresden-3 fault tree.

```
HPIS  1000
+1000   5100110021003103610?1
+1012   4  31  32  33  34
+1004   4  35  36  37  38
X1001   2101?1004
+1020   5  99 101 102 105 106
+1019   4 107 108 110 111
X1011   2102010?9
+1062   6 180  88  89  90  91  97
+1063   9  92  93  94  95  96  97  98 192 181
X1061   2106?1063
+1014   31061  17  18
+1009   3101?10531014
+1005   310091046   4
+1002   41005   1   2   3
+1015   5  69  70  71 .72 188
+1016   5  73  74  75  76 189
X1010   2101510?6
+1006   61010   5   6   7   8   9
+1007   3  13  15  16
+1017   6  77  79  80  81  78 189
+1018   6  82  84  85  86  83 188
X1011   210171018
+1008   41011  10  11  12
+1002   3100610071008
+1044   6  61  62  63  64 185 191
+1045   6  65  66  67  68 186 190
X1040   210441045
X1041   2  59  60
+1017   210401041
+1041   4 133 134 185 191
+1043   4 135 136 190 186
X1028   210421043
+1039  12  19  20  21  22  23  24  25  26  27  28  29  30
+1036   3103710381039
+1049   7  39  40  41  42  43 183 189
+1050   7  44  45  46  47  48 184 188
X1047   210491050
+1051   7  49  50  51  52  53 183 189
+1052   7  54  55  56  57  58 184 188
X1048   210511052
+1046   210471048
+1058   6 115 116 117 118 119 181
+1057   21060 114
X1056   210571058
+1054   41056 112 113 192
+1060   3 126 127 187
+1059   31060 124 125
+1055   51059 120 121 122 123
X1053   210551054
+1064  101070  14 100 102 104 109 129 130 131 132
+1070   7 144 146 147 148 177 178 179
+1034  101071 142 137 143 138 139 149 150 151 152
+1071  10 145 140 153 154 155 141 156 157 158 159
X1032   210351064
+1026   21032 128
+1027   51034 198 196 192 193
X1022   210271026
+1028   21035 198
+1029   3 192 1921033
X1023   210281029
X1033   210341064
X1024   2 1971033
X1030   210341035
+1031   31064 198 199
X1025   210301031
+1021   4102210231024?025
+1035  111072 160 161 162 163 164 165 166 167 168 169
+1072   9 170 171 172 173 174 175 176 194 195
#
```

Fig. 9. Input data for the HPIS fault tree.

```
CHECK OF INPUT DATA FOR :      HPIS   (PRIMAL)

GATE 1000 OCCURS ON LEFT SIDE BUT NOT ON RIGHT

NO OF DIFFERENT OR - GATES =      49
NO OF DIFFERENT AND - GATES =     19

TOTAL NO OF DIFFERENT GATES =     68
NO OF DIFFERENT EVENTS =         199

EVENTS       1   2   3   4   5   6   7   8   9  10  11  12
REPLICATION  1   1   1   1   1   1   1   1   1   1   1   1

EVENTS      13  14  15  16  17  18  19  20  21  22  23  24
REPLICATION  1   4   1   1   1   1   1   1   1   1   1   1

EVENTS      25  26  27  28  29  30  31  32  33  34  35  36
REPLICATION  1   1   1   1   1   1   1   1   1   1   1   1

EVENTS      37  38  39  40  41  42  43  44  45  46  47  48
REPLICATION  1   1   1   1   1   1   1   1   1   1   1   1

EVENTS      49  50  51  52  53  54  55  56  57  58  59  60
REPLICATION  1   1   1   1   1   1   1   1   1   1   1   1

EVENTS      61  62  63  64  65  66  67  68  69  70  71  72
REPLICATION  1   1   1   1   1   1   1   1   1   1   1   1

EVENTS      73  74  75  76  77  78  79  80  81  82  83  84
REPLICATION  1   1   1   1   1   1   1   1   1   1   1   1

EVENTS      85  86  87  88  89  90  91  92  93  94  95  96
REPLICATION  1   1   1   1   1   1   1   1   1   1   1   1

EVENTS      97  98  99 100 101 102 103 104 105 106 107 108
REPLICATION  1   1   1   4   1   4   1   4   1   1   1   1

EVENTS     109 110 111 112 113 114 115 116 117 118 119 120
REPLICATION  4   1   1   1   1   1   1   1   1   1   1   1

EVENTS     121 122 123 124 125 126 127 128 129 130 131 132
REPLICATION  1   1   1   1   1   2   2   1   4   4   4   4

EVENTS     133 134 135 136 137 138 139 140 141 142 143 144
REPLICATION  1   1   1   1   4   4   4   4   4   4   4   4

EVENTS     145 146 147 148 149 150 151 152 153 154 155 156
REPLICATION  4   4   4   4   4   4   4   4   4   4   4   4

EVENTS     157 158 159 160 161 162 163 164 165 166 167 168
REPLICATION  4   4   4   3   3   3   3   3   3   3   3   3

EVENTS     169 170 171 172 173 174 175 176 177 178 179 180
REPLICATION  3   3   3   3   3   3   3   3   4   4   4   1

EVENTS     181 182 183 184 185 186 187 188 189 190 191 192
REPLICATION  1   1   2   2   2   2   2   4   4   2   2   4

EVENTS     193 194 195 196 197 198 199
REPLICATION  2   3   3   1   1   3   1
```

Fig. 10. TREECH output for the HPIS fault tree.

GATE	BICS	MAX LENGTH	REPLICATION
1000	20511	3	
1001	16	2	1
1002	421	3	1
1003	72	2	1
1004	4	1	1
1005	418	3	1
1006	38	2	1
1007	3	1	1
1008	39	2	1
1009	319	3	1
1010	25	2	1
1011	36	2	1
1012	4	1	1
1013	20	2	1
1014	56	2	1
1015	5	1	1
1016	5	1	1
1017	6	1	1
1018	6	1	1
1019	4	1	1
1020	5	1	1
1021	19937	3	1
1022	7015	3	1
1023	6120	3	1
1024	304	3	1
1025	6498	3	1
1026	305	2	1
1027	23	1	1
1028	20	1	1
1029	306	2	1
1030	361	2	1
1031	18	1	1
1032	304	2	1
1033	304	2	2
1034	19	1	3
1035	19	1	3
1036	65	2	1
1037	37	2	1
1038	16	2	1
1039	12	1	1
1040	36	2	1
1041	1	2	1
1042	4	1	1
1043	4	1	1
1044	6	1	1
1045	6	1	1
1046	98	2	1
1047	49	2	1
1048	49	2	1
1049	7	1	1
1050	7	1	1
105[illegible]	7	1	1
1052	7	1	1
1053	241	3	1
1054	27	2	1
1055	9	1	1
1056	24	2	1
1057	4	1	1
1058	6	1	1
1059	5	1	1
1060	3	1	2
1061	54	2	1
1062	6	1	1
1063	9	1	1
1064	16	1	2
1070	7	1	1
1071	10	1	1
1072	9	1	1

Fig. 10. TREECH output for the HPIS fault tree.

GATE	BICS	MAX LENGTH	REPLICATION
1000	248832	174	
1001	2	4	1
1002	48	43	1
1003	4	22	1
1004	1	4	1
1005	48	40	1
1006	2	10	1
1007	1	3	1
1008	2	9	1
1009	12	25	1
1010	2	5	1
1011	2	6	1
1012	1	4	1
1013	2	5	1
1014	2	11	1
1015	1	5	1
1016	1	5	1
1017	1	6	1
1018	1	6	1
1019	1	4	1
1020	1	5	1
1021	81	82	1
1022	3	23	1
1023	3	21	1
1024	3	19	1
1025	3	19	1
1026	2	20	1
1027	1	23	1
1028	1	20	1
1029	2	21	1
1030	2	19	1
1031	1	18	1
1032	2	19	1
1033	2	19	2
1034	1	19	3
1035	1	19	3
1036	8	23	1
1037	4	7	1
1038	2	4	1
1039	1	12	1
1040	2	6	1
1041	2	1	1
1042	1	4	1
1043	1	4	1
1044	1	6	1
1045	1	6	1
1046	4	14	1
1047	2	7	1
1048	2	7	1
1049	1	7	1
1050	1	7	1
1051	1	7	1
1052	1	7	1
1053	3	9	1
1054	2	9	1
1055	1	9	1
1056	2	6	1
1057	1	4	1
1058	1	6	1
1059	1	5	1
1060	1	3	2
1061	2	9	1
1062	1	6	1
1063	1	9	1
1064	1	16	3
1070	1	7	1
1071	1	10	1
1072	1	9	1

Fig. 11. TREECH output for the Dual of the HPIS fault tree.

```
 R CUTSET
CUTSET OR TIESET: CUTSET
CUTSET OF:
 FILE DSK@: HPIS
SYSTEM: HPIS
HIGHEST ORDER WANTED = 999
 TOP GATE  1000
FACTORIZE
FACTORIZE
FACTORIZE
EVALUATE
MINIMIZE
MINIMIZE
OVERFLOW
REDUCE
OUTPUT


 RESULT OF HPIS

 REDUCED CUTSET:
        1.  SETS OF ORDER  1
       43.  SETS OF ORDER  2
        8.  SETS OF ORDER  3
 ------------
       52.

 EVALUATED CUTSET:
       29   SETS OF ORDER  1
      400.  SETS OF ORDER  2
     7750.  SETS OF ORDER  3
 ------------
     8179.
```

Fig. 12. Teletype output during calculation of minimal cut sets of the HPIS fault tree.

```
CUTSET OF HPIS
    4   -1   999   31   32
    4   -1   998   35   36
    4   -1   997   99  101
    4   -1   996  107  108
    4   -1   995   87   88
    4   -1   994   92   93
    4   -1   993   69   70
    4   -1   992   73   74
    4   -1   991   77   78
    4   -1   990   82   83
    4   -1   989   61   62
    4   -1   988   65   66
    4   -2   987   59   60
    4   -1   986  133  134
    4   -1   985  135  136
    4   -1   984   39   40
    4   -1   983   44   45
    4   -1   982   49   50
    4   -1   981   54   55
    4   -1   980  115  116
    4   -1   979  112  113
    4   -1   978  126  127
    4   -1   977   14  100
    4   -1   976  137  138
    4   -1   975  160  161
    4   -1   974    1    2
    4   -1   973   33   34
    4   -1   972   37   38
    4   -1   971  103  105
    4   -1   970  110  111
    4   -1   969   89   90
    4   -1   968   94   95
    4   -1   967   71   72
    4   -1   966   75   76
    4   -1   965   79   80
    4   -1   964   84   85
    4   -1   963   63   64
    4   -1   962   67   68
    4   -1   961  185  191
    4   -1   960  186  190
    4   -1   959   41   42
    4   -1   958   46   47
    4   -1   957   51   52
    4   -1   956   56   57
    4   -1   955  117  118
    4   -1   954  187  978
    4   -1   953  102  104
    4   -1   952  139  140
    4   -1   951  162  163
    4   -1   950    3    4
    4   -1   949  973  999
    4   -1   948  972  998
    4   -1   947  106  971
    4   -1   946  970  996
    4   -1   945   91  180
    4   -1   944   96   97
    4   -1   943  967  993
    4   -1   942  966  992
    4   -1   941   81  965
    4   -1   940   86  964
    4   -1   939  963  989
    4   -1   938  962  988
    4   -1   937   43  959
    4   -1   936   48  958
    4   -1   935   53  957
    4   -1   934   58  956
    4   -1   933  119  181
    4   -1   932  120  121
    4   -1   931  109  129
    4   -1   930  141  142
    4   -1   929  164  165
    4   -1   928    5    6
    4   -1   927  947  997
    4   -1   926  945  969
    4   -1   925   98  182
    4   -1   924  941  991
    4   -1   923  940  990
    4   -1   922  937  984
    4   -1   921  936  983
    4   -1   920  935  982
    4   -1   919  934  981
    4   -1   918  933  955
    4   -1   917  122  123
    4   -1   916  130  131
    4   -1   915  143  145
    4   -1   914  166  167
    4   -1   913    7    8
    4   -1   912  926  995
    4   -1   911  925  944
    4   -1   910  918  980
    4   -1   909  124  125
    4   -1   908  132  144
    4   -1   907  149  150
    4   -1   906  168  169
    4   -1   905    9   10
    4   -1   904  911  968
    4   -1   903  909  917
    4   -1   902  146  147
    4   -1   901  151  152
    4   -1   900  170  171
    4   -1   899   11   12
    4   -1   898  904  994
    4   -1   897  903  932
    4   -1   896  148  177
    4   -1   895  153  154
    4   -1   894  172  173
    4   -1   893  172  173
    4   -1   892   13   15
    4   -1   891  178  179
    4   -1   890  178  179
    4   -1   889  155  156
    4   -1   888  155  156
    4   -1   887  174  175
    4   -2   886  948  949
    4   -2   885  927  946
    4   -1   884   16   17
    4   -1   883  891  896
    4   -1   882  157  158
    4   -1   881  176  194
    4   -1   880   18   19
    4   -1   879  883  902
    4   -1   878  159  882
    4   -1   877  195  881
    4   -1   876   20   21
    4   -1   875  879  908
    4   -1   874  878  889
    4   -1   873  877  887
    4   -1   872   22   23
    4   -1   871  875  916
    4   -1   870  874  895
    4   -1   869  873  894
```

Fig. 13. Result file for the HPIS fault tree.

```
   4   -1  868   24   25
   4   -1  867  871  931
   4   -1  866  870  901
   4   -1  865  869  900
   4   -1  864   26   27
   4   -1  863  867  953
   4   -1  862  866  907
   4   -1  861  865  906
   4   -1  860   28   29
   4   -1  859  863  977
   4   -1  858  862  915
   4   -1  857  861  914
   4   -1  856   30  860
   4   -1  855  858  930
   4   -1  854  857  929
   4   -1  853  856  864
   4   -1  852  855  952
   4   -1  851  854  951
   4   -1  850  853  868
   4   -1  849  852  976
   4   -1  848  851  975
   4   -1  847  850  872
   4   -1  846  847  876
   4   -1  845  846  880
   4   -1  844  845  884
   4   -1  843  844  885
   4   -1  842  843  886
   4   -1  841  842  892
   4   -1  840  841  899
   4   -1  839  840  905
   4   -1  838  839  913
   4   -1  837  838  928
   4   -1  836  837  950
   4   -1  835  836  974
   4   -1  834  835  987
-833
   1  834
   2  128  192
   2  128  193
   2  128  196
   2  128  198
   2  128  849
   2  183  184
   2  183  188
   2  183  919
   2  183  921
   2  184  189
   2  184  920
   2  184  922
   2  188  189
   2  188  920
   2  188  922
   2  188  924
   2  188  942
   2  189  919
   2  189  921
   2  189  923
   2  189  943
   2  192  198
   2  192  848
   2  192  897
   2  192  912
   2  192  954
   2  193  198
   2  193  848
   2  897  979
   2  898  912
   2  910  954
   2  919  920
   2  921  922
   2  923  924
   2  938  939
   2  938  961
   2  939  960
   2  942  943
   2  954  979
   2  960  961
   2  960  986
   2  961  985
   2  985  986
   3  114  897  910
   3  196  848  859
   3  197  849  859
   3  198  848  849
   3  198  848  859
   3  198  849  859
   3  199  848  849
   3  848  849  859
```

Fig. 13. Result file for the HPIS fault tree.

```
DRESD1
1.1.2.1  .10
2.1.2.0 1.15
3.1.2.9 009.48
4.1.2.9 009.48
5.1.2.100  .114
6.1.1.1  .5
7.1.2.1 .10
8.2.2.9 009.48
9.3.2.9 009.48
10.3.2.1  .5
11.3.2.1 .5
12.3.2.5 .8
13.3.2.1  .5
14.3.2.9 009.48
15.3.2.9 009.48
16.3.2.1 .5
17.3.2.5 .8
18.3.2.1 .5
19.3.2.5 .8
20.3.2.5 .1
21.3.2.5 .8
22.3.2.10 .1
23.3.2.10 .200
24.3.2.1  .5
25.3.2.10  .1
26.3.2.5 .8
27.3.2.9 009.48
28.3.2.9 009.48
29.3.2.1 .5
30.3.2.5 .8
31.3.2.10 .1
32.3.2.5 .8
33.3.2.10 .1
34.3.2.5 .8
35.3.2.5 .1
36.3.2.0 5.8
37.3.2.10 .1
38.3.2.1 .70
39.3.2.10  .10
40.3.2.5  .8
41.3.2.10 .1
42.3.2.5 .8
43.3.2.10 .1
44.3.2.1 .5
45.3.2.9 009.48
46.3.2.9 009.48
47.3.2.1 .5
48.3.2.5 .8
49.3.2.10 .1
50.3.2.5 .8
51.3.2.10 .1
52.3.2.1 .5
53.3.2.9 009.48
54.3.2.9 009.48
55.3.2.1 .5
56.3.2.5 .8
57.3.2.10 .1
0
```

Fig. 14. Failure data for the components of the Dresden-3 system.

```
UNAVAILABILITY CALCULATION FOR FAULT TREE: DRESD3
CALCULATION BASED ON CUTSET

NO OF MINIMAL SETS =          48

          TIME        UNAVAILABILITY

       168.000        0.91181E-09      0.91177E-09
       336.000        0.13236E-07      0.13233E-07
       504.000        0.64278E-07      0.64253E-07
       672.000        0.19802E-06      0.19788E-06
       840.000        0.47404E-06      0.47355E-06
      1008.000        0.96711E-06      0.96568E-06
      1176.000        0.17660E-05      0.17625E-05
      1344.000        0.29737E-05      0.29660E-05
      1512.000        0.47047E-05      0.46894E-05
      1680.000        0.70868E-05      0.70585E-05
      3360.000        0.10227E-03      0.10070E-03
      5040.000        0.47188E-03      0.45615E-03
      6720.000        0.13669E-02      0.12880E-02
      8400.000        0.30724E-02      0.28014E-02
     10079.937        0.58895E-02      0.51565E-02
     11760.002        0.10125E-01      0.84411E-02
     13440.001        0.16087E-01      0.12650E-01
     15119.999        0.24080E-01      0.17669E-01
     16799.998        0.34408E-01      0.23258E-01
```

Fig. 15. Unavailability of the Dresden-3 system. Non-repairable case. Two inclusion-exclusion terms.

```
UNAVAILABILITY CALCULATION FOR FAULT TREE  DRESD3
CALCULATION BASED ON CUTSET

NO OF MINIMAL SETS =          48

          TIME        UNAVAILABILITY

         0.000        0.00000E+00      0.00000E+00
       200.000        0.42631E-10      0.42631E-10
       400.000        0.66085E-10      0.66085E-10
       600.000        0.65433E-10      0.65433E-10
       800.000        0.65638E-10      0.65638E-10
0
```

Fig. 16. Unavailability of the Dresden-3 system. Repairable case. Two inclusion-exclusion terms.

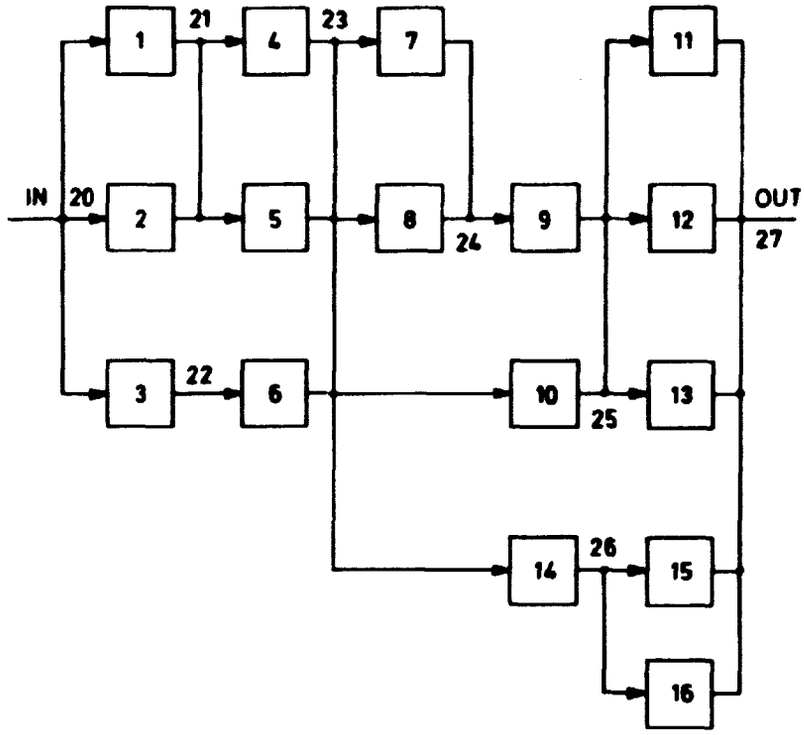


Fig. 17. Uni-directional network (taken from reference 22) used to illustrate the TIENET program.

```
NBBEX2
-1, 20, 21
-2, 20, 21
-3, 20, 22
-4, 21, 23
-5, 21, 23
-6, 22, 23
-7, 23, 24
-8, 23, 24
-9, 24, 25
-10, 23, 25
-11, 25, 27
-12, 25, 27
-13, 25, 27
-14, 23, 26
-15, 26, 27
-16, 26, 27
0
```

Fig. 18. Input data file to TIENET. Data corresponds to the network in fig. 17.

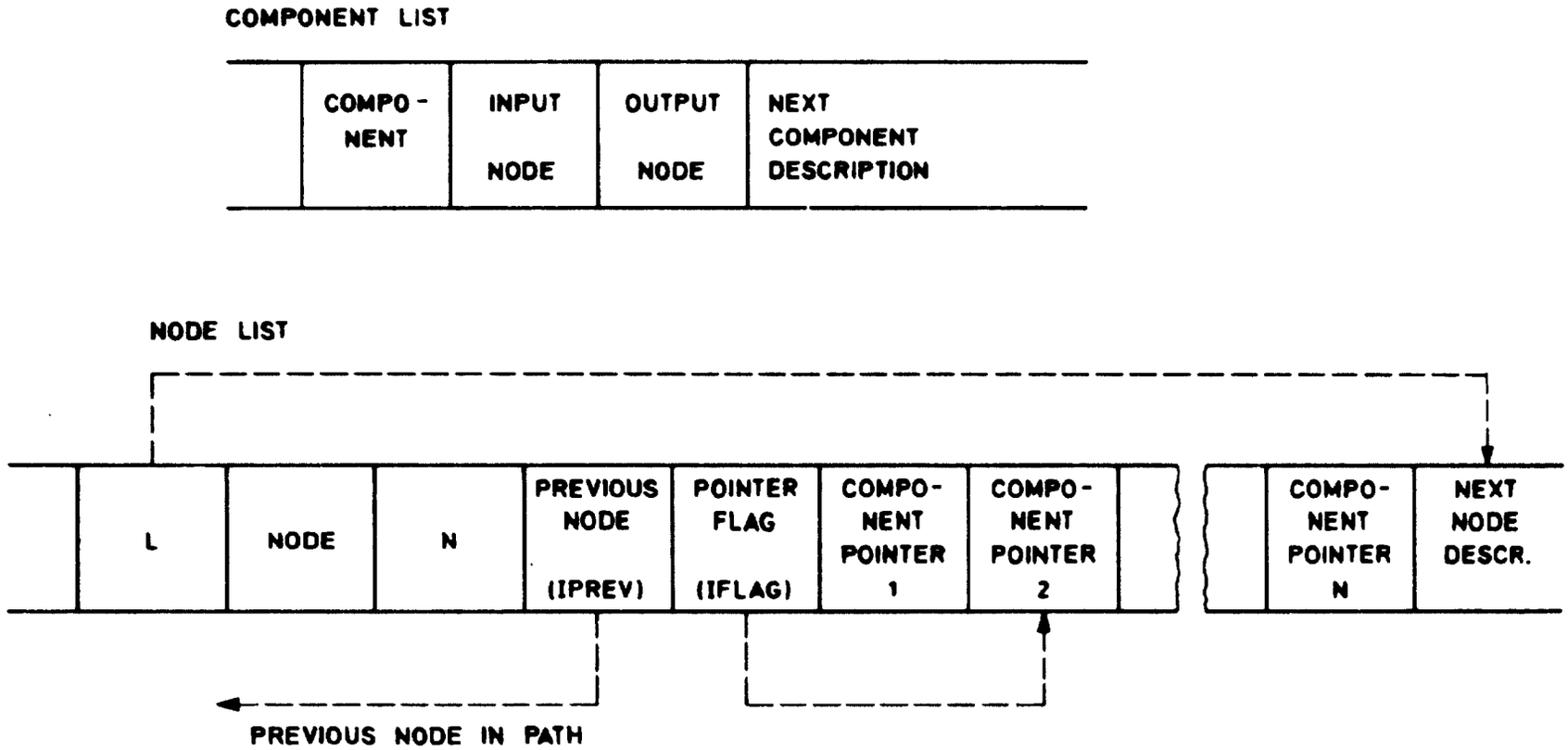


Fig. 19. Internal lists in TIENET.

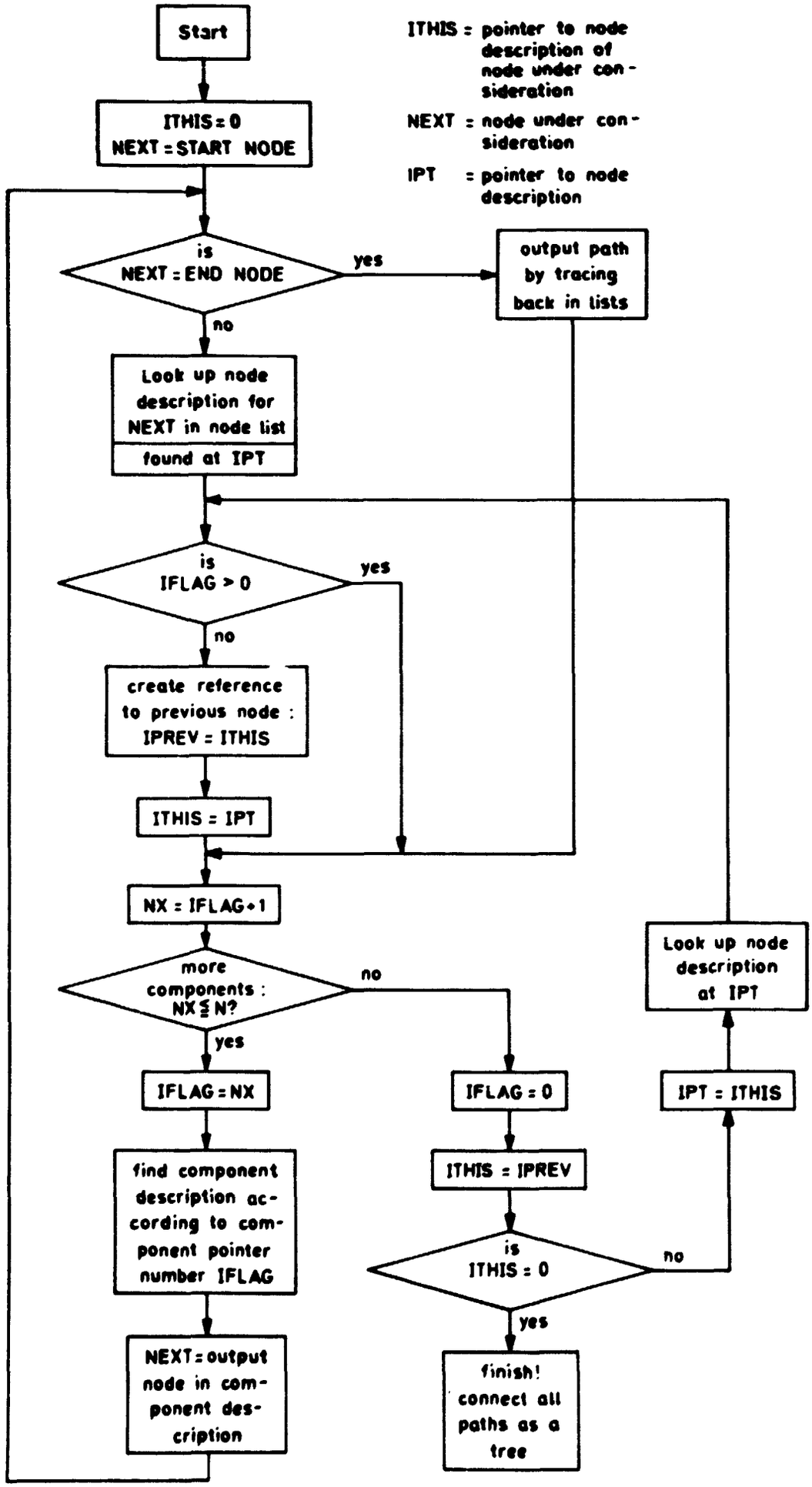


Fig. 20. Flowchart for TIENET.

```
NBBEX2
X1000   11001
+1002   5   1   4   7   9  11
+1003   5   1   4   7   9  12
+1004   5   1   4   7   9  13
+1005   5   1   4   8   9  11
+1006   5   1   4   8   9  12
+1007   5   1   4   8   9  13
+1008   4   1   4  10  11
+1009   4   1   4  10  12
+1010   4   1   4  10  13
+1011   4   1   4  14  15
+1012   4   1   4  14  16
+1013   5   1   5   7   9  11
+1014   5   1   5   7.  9  12
+1015   5   1   5   7   9  13
+1016   5   1   5   8   9  11
+1017   5   1   5   8   9  12
+1018   5   1   5   8   9  13
+1019   4   1   5  10  11
+1020   4   1   5  10  12
+1021   4   1   5  10  13
+1022   4   1   5  14  15
+1023   4   1   5  14  16
+1024   5   2   4   7   9  11
+1025   5   2   4   7   9  12
+1026   5   2   4   7   9  13
+1027   5   2   4   8   9  11
+1028   5   2   4   8   9  12
+1029   5   2   4   8   9  13
+1030   4   2   4  10  11
+1031   4   2   4  10  12
+1032   4   2   4  10  13
+1033   4   2   4  14  15
+1034   4   2   4  14  16
+1035   5   2   5   7   9  11
+1036   5   2   5   7   9  12
+1037   5   2   5   7   9  13
+1038   5   2   5   8   9  11
+1039   5   2   5   8   9  12
+1040   5   2   5   8   9  13
+1041   4   2   5  10  11
+1042   4   2   5  10  12
+1043   4   2   5  10  13
+1044   4   2   5  14  15
+1045   4   2   5  14  16
+1046   5   3   6   7   9  11
+1047   5   3   6   7   9  12
+1048   5   3   6   7   9  13
+1049   5   3   6   8   9  11
+1050   5   3   6   8   9  12
+1051   5   3   6   8   9  13
+1052   4   3   6  10  11
+1053   4   3   6  10  12
+1054   4   3   6  10  13
+1055   4   3   6  14  15
+1056   4   3   6  14  16
X1001  10100210031004100510061007100810091010 1057
X1057  10101110121013101410151016101710181019 1058
X1058  10102010211022102310241025102610271028 1059
X1059  10102910301031103210331034103510361037 1060
X1060  10103810391040104110421043104410451046 1061
X1061  10104710481049105010511052105310541055 1056
$
```

Fig. 21. Output file from TIENET. The data corresponds to the network in fig. 17.

```
 R CUTSET
CUTSET OR TIESET. CUTSET
CUTSET OF
 FILE DSK0: P20T27
SYSTEM  NBBEX2
HIGHEST ORDER WANTED = 999
 TOP GATE:
GATE. 1000 SELECTED AS TOP
FACTORIZE
EVALUATE
MINIMIZE
MINIMIZE
MINIMIZE
MINIMIZE
MINIMIZE
MINIMIZE
MINIMIZE
MINIMIZE
MINIMIZE
MINIMIZE
MINIMIZE
MINIMIZE
MINIMIZE
MINIMIZE
MINIMIZE
OVERFLOW
REDUCE
OUTPUT


 RESULT OF NBBEX2

 REDUCED CUTSET:
        3. SETS OF ORDER  3
        3. SETS OF ORDER  4
        2. SETS OF ORDER  5
 ------------
        8.

 EVALUATED CUTSET:
        5. SETS OF ORDER  3
        3. SETS OF ORDER  4
        2. SETS OF ORDER  5
 ------------
       10


CUTSET OF NBBEX2
    4   -1  999    3    6
 -998
    3    1    2  999
    3    4    5  999
    3    9   10   14
    4    7    8   10   14
    4    9   10   15   16
    4   11   12   13   14
    5    7    8   10   15   16
    5   11   12   13   15   16
```

Fig. 22. Minimal cut sets for the network in fig. 17.

```
 R CUTSET
CUTSET OR TIESET: TIESET
TIESET OF:
 FILE DSK0: P20127
SYSTEM: NBBEX2
HIGHEST ORDER WANTED = 999
 TOP GATE:
GATE: 1000 SELECTED AS TOP
FACTORIZE
EVALUATE
MINIMIZE
OVERFLOW
REDUCE
OUTPUT


 RESULT OF NBBEX2

 REDUCED TIESET:
       5. SETS OF ORDER  3
      26. SETS OF ORDER  4
      24. SETS OF ORDER  5
 ------------
      55.

 EVALUATED TIESET:
      25. SETS OF ORDER  4
      30. SETS OF ORDER  5
 ------------
      55.
```

```
TIESET OF NBBEX2
    4   -2  999    3    6
 -998
    3   10   11  999
    3   10   12  999
    3   10   13  999
    3   14   15  999
    3   14   16  999
    4    1    4   10   11
    4    1    4   10   12
    4    1    4   10   13
    4    1    4   14   15
    4    1    4   14   16
    4    1    5   10   11
    4    1    5   10   12
    4    1    5   10   13
    4    1    5   14   15
    4    1    5   14   16
    4    2    4   10   11
    4    2    4   10   12
    4    2    4   10   13
    4    2    4   14   15
    4    2    4   14   16
    4    2    5   10   11
    4    2    5   10   12
    4    2    5   10   13
    4    2    5   14   15
    4    2    5   14   16
    4    7    9   11  999
    4    7    9   12  999
    4    7    9   13  999
    4    8    9   11  999
    4    8    9   12  999
    4    8    9   13  999
    5    1    4    7    9   11
    5    1    4    7    9   12
    5    1    4    7    9   13
    5    1    4    8    9   11
    5    1    4    8    9   12
    5    1    4    8    9   13
    5    1    5    7    9   11
    5    1    5    7    9   12
    5    1    5    7    9   13
    5    1    5    8    9   11
    5    1    5    8    9   12
    5    1    5    8    9   13
    5    2    4    7    9   11
    5    2    4    7    9   12
    5    2    4    7    9   13
    5    2    4    8    9   11
    5    2    4    8    9   12
    5    2    4    8    9   13
    5    2    5    7    9   11
    5    2    5    7    9   12
    5    2    5    7    9   13
    5    2    5    8    9   11
    5    2    5    8    9   12
    5    2    5    8    9   13
```

Fig. 23. Minimal path sets for the network in fig. 17.

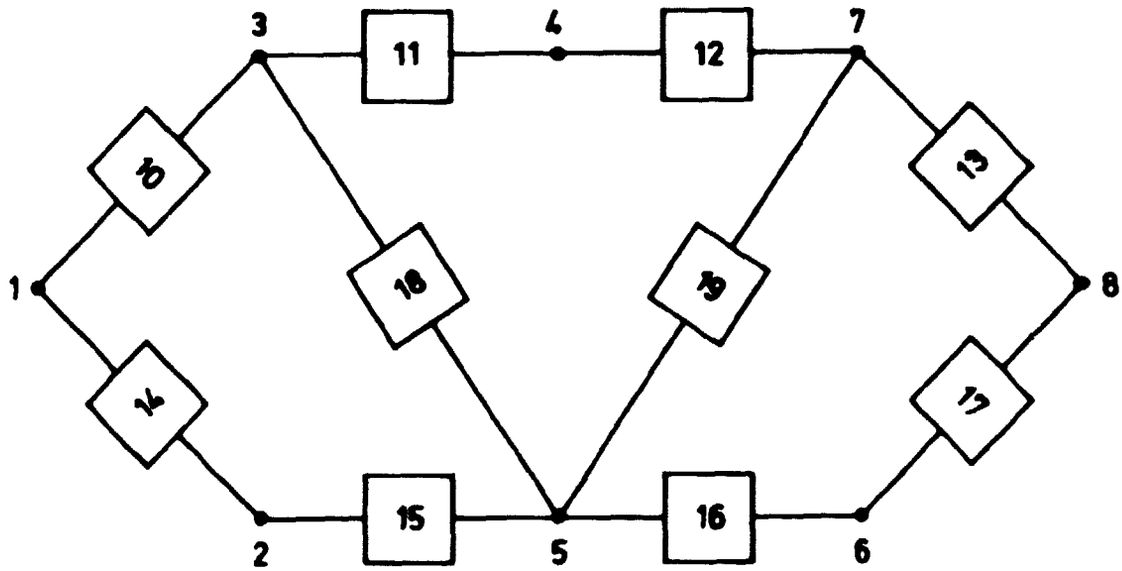


Fig. 24. Bi-directional network (taken from reference 20) used to illustrate the TIENET program.

```
TYPE JBFIG1.DA

JBFIG1
10,1,3
11,3,4
12,4,7
13,7,8
14,1,2
15,2,5
16,5,6
17,6,8
18,3,5
19,5,7
0
```

Fig. 25. TIENET input data for the network in fig. 24.

```
 R TIENET
NETWORK PATHS
OUTPUT NODES,COMPONENTS OR BOTH? NODES
NETWRK FILE
 FILE DSK@ JBFIG1
CALCULATE PATHS FROM NODE 1
TO NODE 8
OUTPUT ON
 FILE DSK@ N01T08

 R TIENET
NETWORK PATHS
OUTPUT NODES,COMPONENTS OR BOTH? COMPONENTS
NETWRK FILE
 FILE DSK@ JBFIG1
CALCULATE PATHS FROM NODE 1
TO NODE 8
OUTPUT ON
 FILE DSK@ C01T08

 R TIENET
NETWORK PATHS
OUTPUT NODES,COMPONENTS OR BOTH? BOTH
NETWRK FILE
 FILE DSK@ JBFIG1
CALCULATE PATHS FROM NODE 1
TO NODE 8
OUTPUT ON
 FILE DSK@ B01T08
```

Fig. 26. Teletype output during three runs of TIENET for the network in fig. 24.

```
 R CUTSET
CUTSET OR TIESET CUTSET
CUTSET OF
 FILE DSK@ N01T08
SYSTEM: JBFIG1
HIGHEST ORDER WANTED = 999
 TOP GATE:
GATE 1000 SELECTED AS TOP
FACTORIZE
EVALUATE
MINIMIZE
MINIMIZE
OVERFLOW
REDUCE
OUTPUT


 RESULT OF JBFIG1

 REDUCED CUTSET:
        1 SETS OF ORDER  1
        5 SETS OF ORDER  2
 ------------
        6.

 EVALUATED CUTSET:
        2 SETS OF ORDER  1
        5 SETS OF ORDER  2
 ------------
        7.


 R CUTEV
EVALUATE COMPLEX EVENTS

 TYPE EVAL DA


CUTSET OF JBFIG1
    1    8
    1    1
    2    2    3
    2    3    5
    2    4    5
    2    5    7
    2    6    7
```

Fig. 27. Minimal cut sets of the network in fig. 24. Nodes only.

```
 R CUTSET
CUTSET OR TIESET: CUTSET
CUTSET OF:
 FILE DSK0. C01T03
SYSTEM: JBFIG1
HIGHEST ORDER WANTED = 999
 TOP GATE:
GATE: 1000 SELECTED AS TOP
FACTORIZE
EVALUATE
MINIMIZE
MINIMIZE
OVERFLOW
REDUCE
OUTPUT


 RESULT OF JBFIG1

 REDUCED CUTSET:
        2. SETS OF ORDER  2
        2. SETS OF ORDER  3
        2. SETS OF ORDER  4
 ------------
        6.

 EVALUATED CUTSET:
        4. SETS OF ORDER  2
        8. SETS OF ORDER  3
        4. SETS OF ORDER  4
 ------------
       16.


 R CUTEV
EVALUATE COMPLEX EVENTS

 TYPE EVAL. DA


CUTSET OF JBFIG1
    2   10   15
    2   10   14
    2   13   17
    2   13   16
    3   12   15   18
    3   12   14   18
    3   11   15   18
    3   11   14   18
    3   12   17   19
    3   12   16   19
    3   11   17   19
    3   11   16   19
    4   10   17   18   19
    4   10   16   18   19
    4   13   15   18   19
    4   13   14   18   19
```

Fig. 28. Minimal cut sets of the network in fig. 24. Components only.

```
 R CUTSET
CUTSET OR TIFSET: CUTSET
CUTSET OF:
 FILE DSK0: B0110S
SYSTEM: JBFIG1
HIGHEST ORDER WANTED = 999
 TOP GATE:
GATE: 1000 SELECTED AS TOP
FACTORIZE
EVALUATE
MINIMIZE
MINIMIZE
OVERFLOW
REDUCE
OUTPUT


 RESULT OF JBFIG1

 REDUCED CUTSET:
        1.  SETS OF ORDER  1
        9.  SETS OF ORDER  2
        4.  SETS OF ORDER  3
        2.  SETS OF ORDER  4
 ------------
       16.

 EVALUATED CUTSET:
        2.  SETS OF ORDER  1
       19.  SETS OF ORDER  2
       24.  SETS OF ORDER  3
        6.  SETS OF ORDER  4
 ------------
       51.


 R CUTEV
EVALUATE COMPLEX EVENTS
```

```
TYPE EVAL DA

CUTSET OF JBFIG1
    1    8
    1    1
    2    3    5
    2    3   14
    2    3   15
    2    2    3
    2    5    7
    2    5   10
    2    5   13
    2    5   11
    2    5   12
    2    4    5
    2    7   16
    2    7   17
    2    6    7
    2   10   14
    2   10   15
    2    2   10
    2   13   16
    2   13   17
    2    6   13
    3    3   16   19
    3    3   17   19
    3    3    6   19
    3    7   14   18
    3    7   15   18
    3    2    7   18
    3   11   14   18
    3   11   15   18
    3    2   11   18
    3   12   14   18
    3    4   14   18
    3   12   15   18
    3    4   15   18
    3    2   12   18
    3    2    4   18
    3   11   16   19
    3   11   17   19
    3    6   11   19
    3   12   16   19
    3    4   16   19
    3   12   17   19
    3    4   17   19
    3    6   12   19
    3    4    6   19
    4   10   16   18   19
    4   10   17   18   19
    4    6   10   18   19
    4   13   14   18   19
    4   13   15   18   19
    4    2   13   18   19
```

Fig. 29. Minimal cut sets of the network in fig. 24. Nodes and components.

```
 R CUTSET
CUTSET OR TIESET: TIESET
TIESET OF:
 FILE DSK0: C01T03
SYSTEM: JBFIG1
HIGHEST ORDER WANTED = 999
 TOP GATE:
GATE: 1000 SELECTED AS TOP
FACTORIZE
EVALUATE
MINIMIZE
OVERFLOW
REDUCE
OUTPUT

 RESULT OF JBFIG1

 REDUCED TIESET:
        1.  SETS OF ORDER  2
        3.  SETS OF ORDER  3
        3.  SETS OF ORDER  4
 ------------
        7.

 EVALUATED TIESET:
        5.  SETS OF ORDER  4
        2.  SETS OF ORDER  6
 ------------
        7.


 R CUTEV
EVALUATE COMPLEX EVENTS

 TYPE EVAL.DA


TIESET OF JBFIG1
    4   14   15   16   17
    4   10   11   12   13
    4   10   16   17   18
    4   13   14   15   19
    4   10   13   18   19
    6   10   11   12   16   17   19
    6   11   12   13   14   15   18
```

Fig. 30. Minimal path sets of the network in fig. 24. Components only.